PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 : G06F 15/06, 15/68, 7/48

A2

(11) International Publication Number: WO 88/ 07722

(43) International Publication Date: 6 October 1988 (06.10.88)

(21) International Application Number: PCT/GB88/00235

(22) International Filing Date: 28 March 1988 (28.03.88)

(31) Priority Application Number: 8707493

(32) Priority Date: 28 March 1987 (28.03.87)

(33) Priority Country: GB

(71) Applicant *(for all designated States except US)*: STONEFIELD SYSTEMS PLC [GB/GB]; Lawson Hunt Industrial Estate, Broadbridge Heath, Horsham, West Sussex (GB).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)* : CONSIDINE, William, Howard [GB/GB]; Haymans Ghyll, Coolham, West Sussex (GB).

(74) Agent: G. F. REDFERN & CO.; Marlborough Lodge, 14 Farncombe Road, Worthing, West Sussex BN11 2BT (GB).

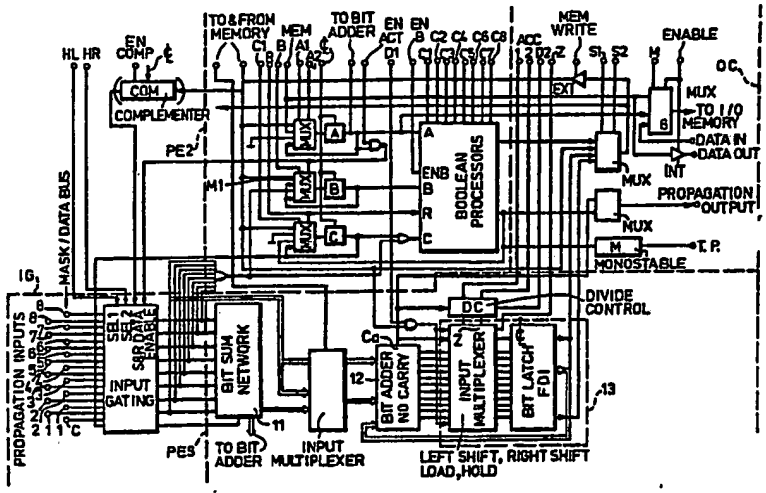
(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent), US.

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: IMPROVEMENTS IN OR RELATING TO CELLULAR ARRAY PROCESSING DEVICES

(57) Abstract

In each processing element of an array an input gate arrangement IG is provided with parallel AND-gates, holding register and ranking network which can be selected to provide outputs (derived from multiple bit neighbouring pixel values) for supply to either a generally conventional processing element PE2 or to a processing circuit PES which together with element PE2 forms an enhanced processing element. A bit summing network (11) in the circuit PES forms a count value signal representing the number of bits of a predetermined logic type supplied to inputs of the network (11), e.g. from the parallel AND-gates in arrangement IG. This count value signal is processed in adder (12) and accumulator (13) for example to form a convolution image value. The ranking network in the arrangement IG can be used to select for further processing a neighbouring pixel value of predetermined rank. Thus the components provided in circuit PES supplement the generally conventional element PE2 to enable simultaneous processing of a plurality of corresponding bits (same bit plane) of neighbouring pixel values so that convolutions and other image processing operations previously requiring inordinate time or hardware outlay are made economically possible at high speed. The components provided in device PES can be used in providing addition, substraction, multiplication and division operations as well as geometric transformations at enhanced speed.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	ML	Mali
AU	Australia	GA	Gabon	MR	Mauritania
BB	Barbados	GB	United Kingdom	MW	Malawi
BE	Belgium	HU	Hungary	NL	Netherlands
BG	Bulgaria	IT	Italy	NO	Norway
BJ	Benin	JP	Japan	RO	Romania
BR	Brazil	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SN	Senegal
CH	Switzerland	LI	Liechtenstein	SU	Soviet Union
CM	Cameroon	LK	Sri Lanka	TD	Chad
DE	Germany, Federal Republic of	LU	Luxembourg	TG	Togo
DK	Denmark	MC	Monaco	US	United States of America
FI	Finland	MG	Madagascar		

IMPROVEMENTS IN OR RELATING TO CELLULAR ARRAY PROCESSING DEVICES

This invention relates to improvements in or relating to cellular array processing devices.

Many areas are emerging that require processing of images in real time. More inspection is required of items on fast production lines (food, mechanical components, printed circuit boards, etc.), especially with increasing quality control standards and the desire for 100% inspection. Automated and robotic assembly requires vision sensor feedback on the position, shape, texture etc. of components, tools, etc. Defence systems need to be able to identify moving targets quickly. All these applications require a fast image processor.

A picture can be represented in digital equipment by a regular grid of squares (called pixels), say 128 units wide by 128 units high. Hence the picture is composed of approximately 16,000 squares. Each pixel has a number associated with it which gives the brightness of that particular point in the picture. The concept used in cellular array processors is to have one relatively simple processor per pixel. That means over 16,000 processors are required for a 128 x 128 machine. The processors sit in a grid type array and each processor can communicate directly to its eight neighbours (Fig. 1). The processors are specifically designed for image manipulation tasks and should not be compared with the typical general purpose microprocessors used in desktop microcomputers etc.

The control of the cellular array processors array is designed so that all the processors can work simultaneously on an image. This parallel working is the key to the speed of the cellular array processors image processing.

Typically the cellular array processors can be used for finding edges, measuring shape, area, perimeters etc. It will enhance poor images by reducing noise, enhancing particular features etc. It will look for particular features such as cracks and edges, and allow texture analysis. It will calculate the shape, centre of gravity etc. for moving objects. While all these processes can be done with a general purpose microcomputer and some extra hardware and programming for, say, 5-10,000, the cellular array processors can be made 100 or more times faster. Hence, instead of inspecting, say, 60 biscuits per minute for shape, size etc. on a production line, the cellular array processor can look at 6,000 per minute - much closer to the real industrial needs. Obviously, in defence applications the ability to calculate rapidly in quickly changing situations is even more important.

The array of single bit processing elements (PE's) is connected together in such a way that each element can access all the data of its eight immediate neighbours. Each processing element has three single bit registers, A,B and C, as well as an additional 32 or 128 single bit locations for image storage. A plane is effectively the natural data unit on the cellular image processor machine, just as an integer is on a serial one. The array, which is driven by the

controller, concerns itself with the manipulation of groups of contiguous planes (denoted bitstacks) to perform some predefined function.

Fig. 2 illustrates the layout of a single array processing element (PE), of which there would be 1024 in a thirty two square (32 x 32) array. The PE's are essentially the heart of the machine, all working together under the guidance of the controller to manipulate planes of bits. With reference to the diagram the constituent parts are described below..

An A register 1, which can be accessed externally as well as from internal sources, provides directly one of the inputs 2 to a boolean processor 3.

A B register 4 can only be loaded from image store registers 5 and depending upon the instruction can contribute to a 'P' input 5 of the boolean processor.

A C or carry register 7 which derives its value from the propagation output and can, depending upon the instruction, contribute to the 'P' input signals. When enabled the C register 7 is loaded simultaneously with the storing of the output result. It ensures that the carry register contains a result from the last operation.

There may be typically thirty two or 128 image storage registers 5. The results of boolean operations on the two inputs 2,6 are stored here.

All the interconnections derived from the eight nearest neighbours and the carry plane can be enabled via input gates. Enabled signals are then combined via an OR-gate G, generating a signal at 'T' which contributes to the 'P' input signal.

The processor has two inputs 2,6 and two outputs 8,9 where each of the outputs 8,9 can be set to any boolean function of the two inputs. The D output 8 for example could be set to equal ($\bar{A}$ and P). The D output gets clocked into the image storage registers 5 when a process-and-store instruction is executed. The N output provides the propagation signal to the surrounding processors as well as to the carry register.

As well as enabling registers etc., the logic enables the processing element to act in a full adder capacity. This is used extensively in bitstack manipulations such as addition routines.

There are a variety of operations allowable on the array which can be split into a number of categories:

Single input operations, where the user is typically only interested in the A input. An example would be that of inverting a plane, i.e. setting the D output to be $\bar{A}$. The B,C registers and the interconnection inputs are unused.

Double input operations which can either be propagating or non-propagating. The operator is interested in both boolean inputs, the P input being derived either from the B or C plane (in the non propagating case) or from the propagation logic. Examples of two such input operations include addition of bitstacks and shifting.

Propagation effects can either be local or global. In the local case the propagating output is not dependent on the propagating input. As a result, a pixel cannot influence the result of another which does not lie in its immediate neighbourhood. This is untrue for global propagating where dependencies can span the array by virtue of letting the propagation output be dependent on the propagating input.

These types of processor cells are ideal for processing black and white images where each pixel of the image consists only of one bit. These bits can then be passed from processor to processor, processed and passed on where necessary. In this way a significant amount of processing can be achieved without returning the pixel values to the storage registers or loading new pixel values from the storage registers.

However, if images with varying shades of grey (grey-scale images) are to be processed, where each pixel is represented by more than one bit, typically 8 bits being used to define 256 shades of grey, only one bit of each pixel can be handled in turn. For instance one simple operation is the addition of two images. If the pixels of these images each consist of 8 bits, then they will be stored in 16 bits of the image storage registers. To accomplish the addition, the least significant bit of one image is loaded into the A register 1, the least significant bit of the other image is loaded into the B register 4, the processor is configured to perform an add operation and the sum of these two bits is returned to the image storage registers 5 as the least significant bit of the sum image. The C

register 7 will then contain the carry output. The next least significant bits are loaded into the A and B registers 1,4, and the add operation performed on these and the carry which has been stored in the C register 7. In this way, eight successive operations each consisting of two transfers from the image storage registers 5 and one transfer to the image storage registers 5 are required to achieve the addition of two 8 bit images.

More complex operations on images frequently require that two images, or an image and a group of coefficients, must be multiplied together. The multiplication of an 8 bit image and an 8 bit coefficient therefore involves eight 8 bit additions, which is 8 x 8 = 64 single bit additions, which is 128 x 3 = 384 transfers to and from the image storage registers 5. Operations on images with varying shades of grey can therefore become quite protracted, even with a large array of simple processors.

Modern integrated circuits will permit the multiplication of two 8 bit numbers in times of the order of 100 nanoseconds. Using an array with a clock period of 100 nanoseconds the 384 operations described will still take 38.4 microseconds. Even if a thousand pixels are being processed at once by a thousand processors this is still 38.4 nanoseconds per pixel which is not much better than that achieved by a single integrated circuit multiplier. It is therefore important that cell structures be found which can carry out operations which include multiplicatons and divisions more efficiently.

One of the more common and powerful image processing operations is termed a convolution. In its simplest form a value is computed at the location of each pixel which consists of the sum of the products of nine coefficients and the nine pixel values of the pixel itself and its eight nearest neighbours. This operation involves nine multiplications and nine additions which will take approximately 3,500 microseconds using the technique described above. Many useful convolutions involve not just the eight nearest neighbours but a larger neighbourhood which may be 5 x 5 or 7 x 7 or even 15 x 15. Operations on larger convolutions therefore become quite slow, and even with larger arrays still not of industrially useful speeds.

It is an object of this invention to provide devices which will carry out convolutions and other common image processing operations at higher speeds than the known arrangements described above.

According to a first aspect of this invention there is provided a cellular array processor comprising an array of processing elements each, as local element, being connected to two or more respective processing elements of the array to form a neighbourhood and capable of providing, as local element, simultaneous processing of corresponding bits of multiple bit values held by all the processing elements of the neighbourhood.

According to a second aspect of this invention there is provided a cellular array processing device comprising means for

summing the nth significant bits of the nth bit plane of a number of selected pixel binary values, means for comparing the sum so obtained with a required rank value, means for masking either ones or zeros out of the bits of the nth bit plane in dependence on said comparison result and means for varying the value of n in a predetermined manner relative to the most significant bit where n is an integer, to process each bit plane in turn whereby the pixel of required rank is selected.

Embodiments of this invention will now be described, by way of example, with reference to the accompanying drawings in which:-

Fig. 1 is a schematic diagram illustrating how a pixel relates to its eight nearest neighbours in an image pixel array;

Fig. 2 is a block circuit diagram of a processing element for use in a cellular array processing device;

Fig. 3 is a schematic block circuit diagram of a portion of a cellular array processing device embodying this invention;

Fig. 4 is a block circuit diagram of an enhanced processing element forming part of the processor portion shown in Fig. 3;

Fig. 5 is a diagram illustrating the stages in the convolution operation of a cellular array processing device embodying this invention;

Fig. 6 is a block circuit diagram of a portion of the cellular array processing element shown in Fig. 4 arranged to operate in accordance with the method illustrated in Fig. 5; and

Figs. 7 to 10 are block circuit diagrams of portions of the cellular array processing element shown in Fig. 4 arranged to carry out other common image processing operations.

Referring to Fig. 3, there is shown schematically the general layout of the enhanced processing elements EPE of a cellular array embodying this invention. The entire array is subject to control by means of logic unit CL which supplies control signals and data via a parallel bus B (which could instead be a serial bus) to an input gate arrangement IG which is also connected to receive data from the neighbours and from local memory LM of the processing element. The local memory LM is arranged for receiving data from output circuit OC of the element EPE. The actual processing arrangement comprises a combination of a processing element PE2 generally similar to that shown in Fig. 2 with a supplementary processing circuit PES. The latter is arranged to provide the various functions described below with reference to Figs. 5 to 10.

Referring to Fig. 4 there is shown in more detail the composite processing element which incorporates all the features described below and shown in Figs. 6,7,8,9 and 10. The element PE2 shares the input gate arrangement IG (and in fact the nine AND-gates 10 and ranking network shown in Fig. 9 which are included in the arrangement IG) with the circuit PES. The details of the element PE2 are not of immediate relevance to this invention which is concerned with the way in which the input gate arrangement IG has been modified and combined with the circuit PES which includes bit sum network 11, adder 12 and accumulator 13 referred to below as well as multiplexer M1 and divide control unit DC. Complementer COM required in the arrangement shown in Fig. 6 is also provided. The

accumulator 13 is shown split into its separate multiplexer 213 for selecting between the hold, load, right and left shift operations and its latches 313. A monostable M in the output circuit OC detects the cessation of activity in the element.

The adder 12 and accumulator 13 shown in Fig. 4 are actually an 11-bit adder and 12-bit accumulator respectively to enable processing for neighbourhoods larger than 3 x 3 although in all the following examples reference will only be made to 3 x 3 neighbourhoods and processing of 8-bit values (8 bit planes, although the invention has application to neighbourhoods of three or more processing elements (one local element and two or more neighbours) utilising pixel or other values of 2 or more bits herein referred to as multiple-bit values). Thus the adder 12 and accumulator 13 will be described as 8-bit and 9-bit devices respectively since these are adequate for 3 x 3 neighbourhoods. Clearly these "reduced" capacity functions can be achieved by appropriate software control using the full-size devices.

While the processing element shown in Fig. 4 comprises all the features required for the various functions described below it will be appreciated that one or more of the functions may not be required, allowing omission of some of the components such as the divide control unit DC and complementer COM.

It will be noted from Fig. 2 that the signals from the nearest neighbour pixels, the interconnection inputs, are each controlled by a control signal 1 to 8. When carrying out the

convolutions in the manner described above, each nearest neighbour is processed in turn, and therefore one of these control inputs will be selected at a time. This is not an efficient use of the logic of the processor structure. It would be much more efficient if the respective bits of all nine pixels could be processed simultaneously using the nine control inputs 1 to 8 and a 9th for the control of the local pixel to supply the corresponding bits of the nine-bit coefficient to be used in calculating the convolution.

Referring to Fig. 5 there are illustrated the 64 terms of the convolution arranged in 15 groups whose maximum possible values are indicated in the right-hand column and these maximum values occur when all the binary A values and binary N coefficient values have logic value "1".

Fig. 3 is derived by the following argument:-

Suppose the nine coefficients are N_0 to N_8 (N_x where x is an integer from 0 to 8), the subscript denoting the relationship to the central pixel (see Fig. 1).

If the value of the central pixel is A_0 and the values of the adjacent pixels are A_1 to A_8 (A_x where x is an integer from 0 to 8), then the value for the convolution at pixel 0 is, by definition:-

$$C = A_0 N_0 + A_1 N_1 + A_2 N_2 + A_3 N_3 + A_4 N_4 + A_5 N_5 + A_6 N_6 + A_7 N_7 + A_8 N_8$$

This calculation is carried out for every pixel in the image, the sum being stored at the local processing element in each case. The result is a 'convolved' image.

Take one of the nine products, say $A_0 N_0$ and expand each of the values into its binary coefficients (A_x^i and N_x^i each having

value 0 or 1) using a superscript 1,j to denote the significance of the coefficient.

$$A_0 N_0 = (A_0^0 + 2A_0^1 + 4A_0^2 + 8A_0^3 + 16A_0^4 + 32A_0^5 + 64A_0^6 + 128A_0^7)$$

$$X (N_0^0 + 2N_0^1 + 4N_0^2 + 8N_0^3 + 16N_0^4 + 32N_0^5 + 64N_0^6 + 128N_0^7)$$

$$= A_0^0 N_0^0 + 2 (A_0^0 N_0^1 + A_0^1 N_0^0) + 4 (A_0^0 N_0^2 + A_0^1 N_0^1 \times A_0^2 N_0^0) + ---$$

and so on.

The value of the convolution is therefore the sum of the expanded terms for all the nine pixels, i.e:

$$\sum_{x=0}^{8} A_x^0 N_x^0 + 2 (\sum_{x=0}^{8} A_x^0 N_x^1 + \sum_{x=0}^{8} A_x^1 N_x^0)$$ and so on.

Note that a binary product such as A.N is the same as an AND operation:-

$$0 \times 0 = 0$$

$$1 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 1 = 1$$

The convolution can therefore be carried out by summing the binary products, starting with the least significant bit.

Each processing element is therefore arranged to AND-link the propagating inputs with the mask, and sums the number of "1" bits resulting and adds this sum to a cumulative total, shifting this convolution total once between each group of products, and storing or discarding the least signficant bit.

Referring to Fig. 4, a network required to accomplish this operation consists of nine AND-gates 10 to AND-link the mask (coefficient values) with the propagation signals (i.e. produce the

products $A_x^i N_x^j$), a bit summing network 11 with nine inputs giving a binary output of the number of input "1" bits, an eight bit adder 12 and a nine bit load and shift accumulator 13..

The bit summing network 11 comprises three stages, a first, input-side, code converter stage, a second, adder stage and a third, output-side, adder stage. The number of "1" bits from the outputs of the AND-gates 10 is determined in code converters 111 of the first stage and provided as a two digit binary value to adders 112 of the second stage which provide three digit binary sum values to adder 113 of the third stage. The final four-digit binary sum output from the third stage represents the total number of "1" values from the outputs of the AND-gates 10 and has a maximum value of 8 which together with the maximum possible "1" value for the local pixel product received at carry input Ca of the 8-bit adder 12 gives a maximum value of 9 for each summation product or term $\Sigma A_x^i N_x^j$ as indicated in the right-hand column of Fig. 3. The 64 terms require 64 cycles to complete summation processing.

Thus of the nine two-input AND-gates 10, one input comes from one of the neighbours of the local element and the other from the parallel bus B which goes to all elements of the array and which bus is used, in this case, to supply the coefficient values N_x^j. The outputs ($A_x^i N_x^j$) of these AND-gates go to the four code converters 111 whose outputs are binary numbers whose value is equal to the number of "1" bits from the AND-gates 10. For convenience the ninth bit goes to the carry input of the bit adder 12. The other 8 outputs

of the AND-gates 10 pass through the code converters 111 whose outputs are two-bit binary values, corresponding to the number of "1" bits at their inputs. The resulting four two-bit binary numbers are summed in the adders 112 to give two three-bit binary numbers which are again summed in adder 113 to give a single four-bit binary number. The output of this last adder 113 goes to the least significant four inputs of one input port of an 8-bit adder 12. The most significant four inputs of the adder 12 are held at binary value "0". The output of this adder 12 goes to the 9-bit accumulator 13 (requiring only 8-bit capacity here), capable of holding, loading and left and right shift operations. The 8-bit output of this accumulator 13 goes to inputs of the other input port of the 8-bit adder 12.

The signals from the neighbours are the A_X^i values in Fig. 3 and the signals from the broadcast databus B are the N_X^j values. On each clock cycle of the network shown in Fig. 4, therefore, the value added to the cumulative total in the accumulator 13 is one of the 64 summation terms $\Sigma A_X^i N_X^j$ in Fig. 3. On each cycle the respective A_X value is loaded into each processing element from local memory LM of that element and therefore propagated to the neighbouring elements. The correct N_X^j value is placed upon the broadcast data bus by the logic CL (Fig. 3) controlling the processor array. At the end of summing each group (groups 1 to 15 in Fig. 3) of terms the least significant bit is shifted out of the accumulator 13 into the local memory LM of the local processing element, and becomes part of the end result of the computation. This shifting achieves the

increase (by a factor of 2 in each case) in significance expressed mathematically by the numerical value (1,2,4,... to 16384) preceding each group in Fig. 3. Additional cycles of the network shown in Fig. 4 will be required after the end of the summation computation itself, in order to shift the accumulated value remaining in the accumulator 13 into the local memory LM.

This processor structure will therefore carry out a 3 by 3 convolution in approximately 80 cycles which, operating at 10 megacycles, will produce a convolution in 8 microseconds.

The processor can be expanded to handle more than nine inputs to provide even faster operation. However it is almost as efficient to carry out larger convolutions in groups of nine pixels, computing the convolution for the first nine, then shifting another group of nine into the immediate locality of the central pixel and performing a second convolution adding the total from the previous convolution into the second convolution. A 15 by 15 convolution can then be carried out as 25 successive 3 by 3 convolutions taking only 200 microseconds.

The same adder 12 and accumulator 13 can be used, with the addition of an 8 bit holding register 14 to perform multiplication and division operations as shown in Figs. 7 and 8 respectively.

Referring to Fig. 7 there is shown how the input gate arrangement IG and the supplementary processing circuit can be used

to provide a local multiplier circuit comprising an 8-bit holding register 14 with its output connected via 8 of the AND-gates 10 to 8 of the inputs of one input port of the 8-bit adder 12. The outputs of the adder 12, including the carry, are connected to the inputs of the 9-bit accumulator 13, which will hold, load and shift left and right. The least significant 8 outputs of the accumulator 13 go to the 8 inputs of the other input port of the adder 12.

Data enters the network either from the parallel (or possibly serial) data bus which goes to all elements of the processor if the multiplicand is a constant common to all the processing elements of the processor, or serially from the local memory LM of the processing element if the multiplicand is a local variable. If the data is the multiplicand it goes to the 8-bit shift/load register, and if it is a multiplier it goes, a bit at a time, to latch 1. Data leaves the multiplier circuit serially from the output of the least significant bit of the accumulator.

Multiplication is achieved by the conventional binary shift and add procedure as shown diagrammatically at the bottom of Fig. 7. The complete multiplication of an 8-bit multiplicand by an 8-bit multiplier takes 16 cycles. On the first 8 of these the value in the 8-bit register 14 is added to the cumulative total in the accumulator 13 if the current bit of the multiplier in latch 15 is 1, and 0 is added if the current bit of the multiplier is zero. On the first cycle, the least significant bit of the multiplier is used, and at the end of the cycle the least significant bit of the result is

shifted out of the accumulator, so that the accumulated total is shifted down one place. These additions take place on the first 8 cycles, and the last 8 cycles are occupied by shifting the most significant bits of the product out of the accumulator 13.

Referring to Fig. 8 there is shown a local division circuit comprising the load and shift register 14 whose inputs are obtained from the broadcast data bus B as the complement of a constant divisor common to all elements, or from the local memory via the complementer COM if the divisor is a local variable. The outputs of this register 14 go to one input port of the 8-bit adder 12, the output of which goes to the hold, load, left and right shift accumulator 13 (which only requires 8-bit capacity here as in Fig. 6). The 8-bit output of the accumulator 13 goes to the other input port of the 8-bit adder 12. A shift input at the least significant end of the accumulator 13 also comes from the local memory. The carry output of the adder 12 is used to control the accumulator 13 and, as the result, is output from the circuit to the local memory. The accumulator 13 is shown in Fig. 8 as including the divider control unit DC.

In operation the accumulator 13 is cleared and the dividend is shifted from the local memory into the least significant end of the accumulator 13, starting with the most significant bit. The complement of the divisor is loaded from the memory into register 14 via the complementer COM or directly from the broadcast data bus B. The sequence of the operations is shown diagramatically at the bottom of Fig. 8. The complement of the divisor is added in adder 12 the

content of the accumulator. In other words the divisor is subtracted from the content of the accumulator. The carry bit of the adder 12 is then the most significant bit of the quotient, since it determines whether there has been a borrow from the most significant position of the subtraction operation. This value also controls the subsequent behaviour of the accumulator 13. If there is no borrow, the output of the adder 12 is loaded into the accumulator 13 since its value is the difference between the divisor and the content of the accumulator. If the carry is 1, there is a borrow, which means that the content of the accumulator is less than the value of the divisor and, therefore, the content of the accumulator is simply shifted one place towards the most significant end and the next most significant bit of the dividend is loaded from the local memory. In this operation 16 cycles will produce a complete 16-bit quotient.

A further common image processing operation is that of ranking. In its simplest form this involves choosing which of the nine pixels in the 3 by 3 mask has the n-th highest value where n lies between 0 and 8. This is not a linear mathematical operation, but can also be performed very fast by means of the proposed processing element structure as shown in Fig. 9.

Referring to Fig. 9, apart from the bit summing network 11 of the processing circuit PES, the ranking network uses a 9 bit mask unit, one bit of the mask for each of the nine propagating inputs. Each mask element of the mask unit consists of a flip-flop 15, an EXCLUSIVE-OR-gate 16 and an AND-gate 17.

At the start of the operation, all flip-flops 15 of the mask unit are set. During the ranking operation the flip-flops 15 of the mask element are progressively cleared using the carry signal from the accumulator 13 until at the end of the operation one (or more in the case where two or more pixels of the required rank are identical) flip-flop 15 of the mask unit remains set. Since the flip-flops of the mask unit, once clear, block the corresponding propagating input at AND-gate 10 forcing the input to zero, the pixel of the required rank may then be shifted into the local memory via the relevant unblocked AND-gate 10.

The ranking operation is performed by serial binary search, starting with the most significant bit. The rank is counted from the highest value pixel.

Before the binary search commences, the value (-RANK-1) where RANK is the value of the required rank, is loaded from the broadcast data bus B through the adder 12 into the accumulator 13 which has previously been cleared. Here only 4-bit capacity is required from the adder 12 and accumulator 13.

The most significant bits of all the pixels are loaded from memory, and propagated (distributed to the processing elements in accordance with the predetermined neighbourhhod arrangement - in the present case, neighbourhoods each comprising 3 x 3 cells or processing elements). The nine bits entering each cell are summed, and the sum is compared in adder 12 with the rank value. If the sum

is equal to or greater than the rank value, then the pixel of the required rank must be one of those with a 1 in the most significant bit, so those with a 0 in this bit are masked off for all succeeding operations. If the sum is less than the rank value, then those with a 1 are masked off, e.g:

Rank value to be selected = 5

Propagating inputs	0 0 1 1 0 1 1 0 1
Required input must be of these	X X X X X
Therefore mask off these	- - - -

If zeros have been masked, the rank value is retained and the operation repeated with the second most significant bit. In the above example, the required pixel is the 5th rank among those pixels selected as having 1 in the most significant bit.

If ones have been masked, then the number of ones is subtracted from the rank value, e.g:

Rank values to be selected = 7

Propagation inputs	0 0 1 1 0 1 1 0 1
Required input must be one of these	X X X X
Therefore mask off these	- - - - -

The required pixel is now 7 - 5 = 2, i.e. the second rank of those selected as having 0 in the most significant bit.

This operation is carried out on all bit planes in succession. The only pixel remaining unmasked at the end of the operation is that of the required rank. The pixel of the required rank can then be shifted into the local memory of the local processing element via the unmasked input.

Another common class of operation is that of geometric transformation, where each pixel of the image is moved to a new position. An example of this is rotation. In parallel arrays, images can be transferred by moving each pixel to one of its nearest neighbours. For simple shifting all the pixels move in the same direction and therefore the direction of motion for all pixels can be broadcast throughout the array. For more complex transformation, such as rotation, each pixel may move in a different direction.

Referring to Fig. 8 there is shown a local directionality circuit comprising an 8-bit latch 19 and eight of the AND-gates 10 for selecting one or more signals from the neighbouring elements, an eight input OR-gate (which is the OR-gate G shown in Fig. 2) to combine the outputs of the AND-gates 10, a multiplexer M to select the local or propagated value, the latch 1 to hold the propagating signal, and the latch 4 to hold the bit read from the local memory that controls the selection of propagated or local data for return to the local memory.

The direction of motion of each pixel may be defined by a direction value stored in the local memory at each location. To make use of this, the 8 bit stored direction value is loaded into a shift register whose outputs are used to select from which of the nearest neighbours the particular pixel is to be loaded (Fig. 10). Using 8 bits for the direction value allows pixels to be merged if required. If only one nearest neighbour was to be selected, a three bit stored direction value would be sufficient.

In order to achieve a transformation such as rotation, where different parts of the image may move at different rates, it is also necessary to have a single bit which controls the loading of the memory or register in addition to the 8 bits which control the direction. An image rotation can then be achieved by a single 8 bit mask which controls the progress of each pixel and a succession of single bit pixels which define those pixels that are to be moved at any stage.

It will be appreciated that while the functions of multiplication, division and pixel displacement described above with reference to Figs, 7, 8 and 10 are carried out in a generally conventional manner, they are achieved with low hardware outlay with substantial gain in speed by using the common components represented by the input gate arrangement IG (including AND-gates 10), the bit sum adder 12 and accumulator 13 which components, when combined with the bit summing network 11, provide a very powerful and fast apparatus for carrying out image processing functions such as convolution and ranking as described with reference to Figs. 5, 6 and 9.

It will also be appreciated that while the description given above relates to operations involving some or all of the nearest neighbours of each processing element, such operations may be arranged to select neighbours of the processing element which are not immediately adjacent so that in this context "neighbour" is to be considered as any other selected processing element in the array.

Also it is to be understood that some references are used for like components in all the figures. Finally, while the embodiment in the above description uses positive logic, it will be clear that negative logic can be used with appropriate changes in the components.

CLAIMS:

1. A cellular array processor comprising an array of processing elements each as local element, being connected to two or more respective processing elements of the array to form a neighbourhood and capable of providing, as local element, simultaneous processing of corresponding bits of multiple bit values held by all the processing elements of the neighbourhood.

2. A processor according to claim 1, wherein each local processing element is related to the other processing elements of the neighbourhood in identical fashion for all the neighbourhoods.

3. A processor according to claim 1 or claim 2, wherein an input gate arrangement has parallel outputs for a plurality of binary signals derived from corresponding bits of the neighbourhood values, said outputs being connected to inputs of a bit summing device arranged to provide an output count value signal representing the number of bits of predetermined binary type received at the inputs of the bit summing device.

4. A processor according to claim 3, wherein said bit summing device is connected to supply said count value signal to a series arrangement of a bit adder and an accumulator.

5. A processor according to any one of the preceding claims, wherein the input gate arrangement has a plurality of parallel logic gates each arranged to receive a pair of input values at least one of which comprises a neighbourhood value, an output of the gate providing a logic value signal representing the product of the input values.

6. A processor according to claim 5 as appendant to claim 4, wherein each gate is provided with a binary coefficient value as one of the input values and the gates provide product signals for all the processing elements of the neighbourhood including the local processing element and said bit adder and accumulator are arranged to form a total of said products representing a convolution of the neighbourhood values using said coefficient.

7. A processor according to claim 5 as appendant to claim 4, wherein said input gate arrangement includes a mask unit arranged for selecting the neighbourhood values passed to the gates in dependence on a predetermined rank value received by the bit adder.

8. A processor according to claim 7, wherein said mask unit includes a plurality of parallel mask elements each being an exclusive OR-gate connected to a latch circuit and an input of each mask element being arranged to receive the associated neighbourhood value whose output provides one of the input signals for the associated one of the logic gates, the mask unit being arranged to select the associated neighbourhood value for transmission or not to the bit summing network.

9. A processor according to any one of the preceding claims, wherein means are provided for broadcasting a multiple bit value to all the processing elements of the array, said broadcasting means including a bus.

10. A processor according to claim 9, wherein means are provided for carrying out addition, subtraction, multiplication and division

of pairs of multiple bit values one of which can comprise a broadcast value.

11. A processor according to claim 9 or claim 10, wherein the input gate arrangement includes means for receiving and holding neighbourhood values and/or broadcast values.

12. A processor according to any one of the preceding claims, wherein means are provided for carrying out geometric transformations on neighbourhood values.

13. A processor according to claim 5, wherein the outputs of said logic gates are connected to inputs of an OR-gate in said input gate arrangement.

14. A cellular array processor comprising means for summing the nth significant bits of the nth bit plane of a number of selected pixel binary values, means for comparing the sum so obtained with a required rank value, means for masking either ones or zeros out of the bits of the nth bit plane in dependence on said comparison result and means for varying the value of n in a predetermined manner relative to the most significant bit where n is an integer, to process each bit plane in turn whereby the pixel of required rank is selected.

15. A cellular array processor substantially as described herein with refernce to Figs. 3 and 4 alone or in combination with any one or more of Figs. 6 to 10 of the accompanying drawings.

FIG. 1

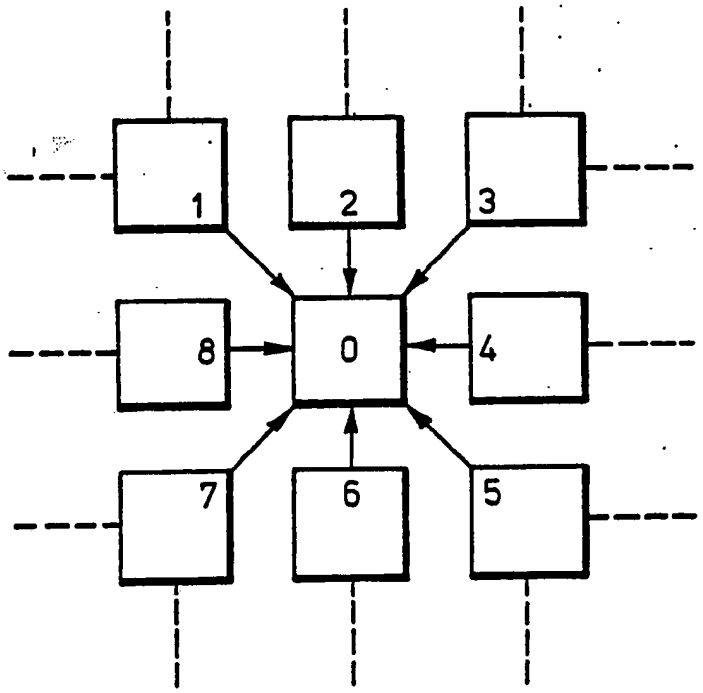
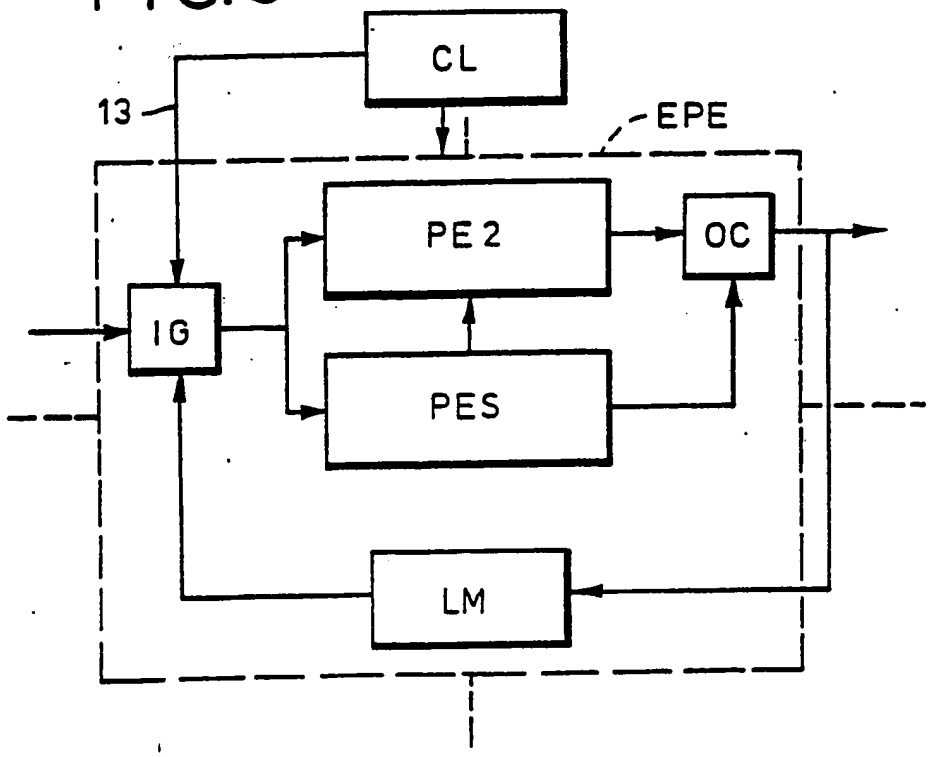


FIG. 3



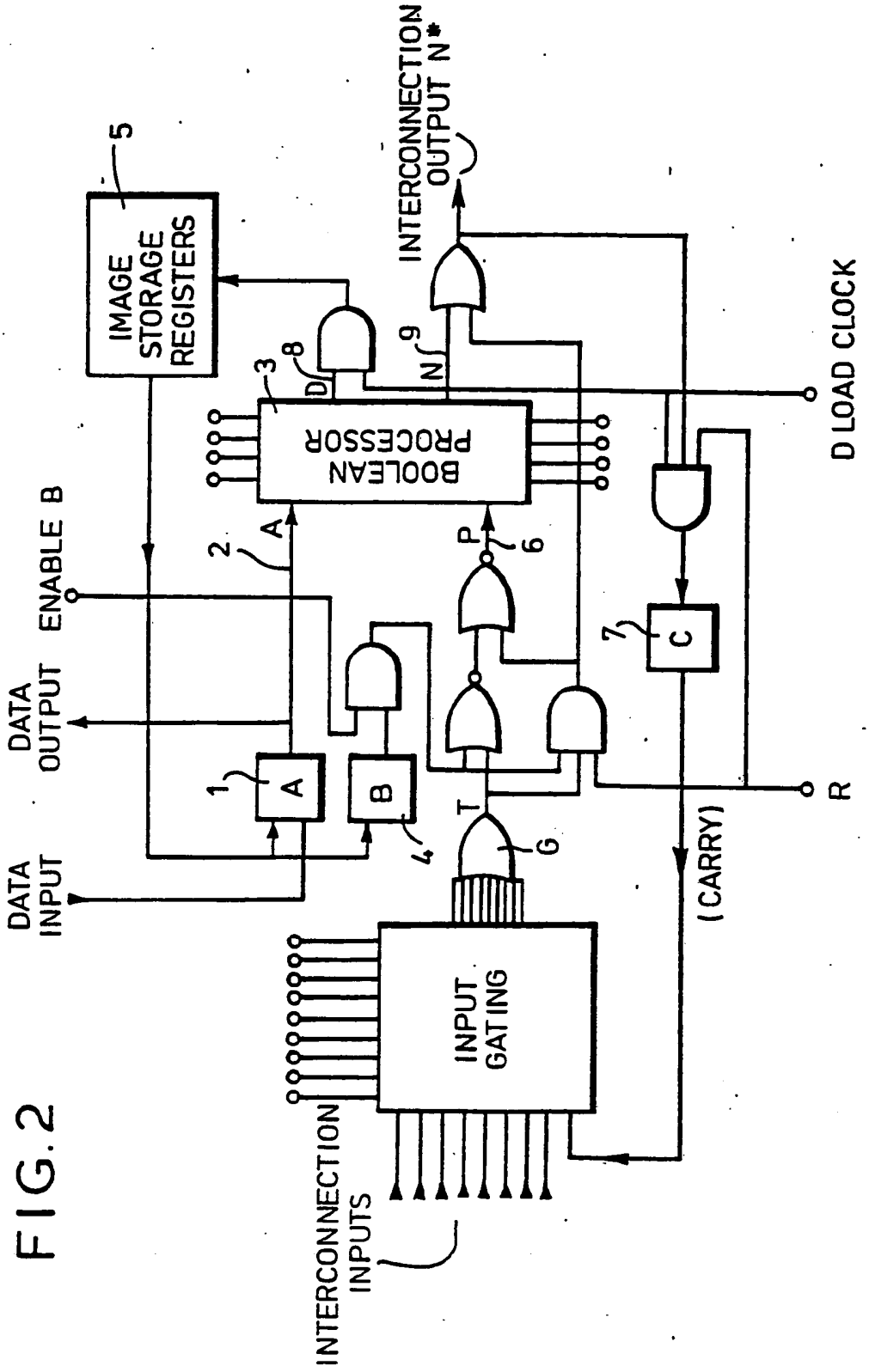


IMAGE STORAGE REGISTERS
5
INTERCONNECTION OUTPUT N*
9
N
8
D
3
BOOLEAN PROCESSOR
D LOAD CLOCK
ENABLE B
2
A
P
6
DATA OUTPUT
1
A
B
4
7
C
T
G
R
DATA INPUT
(CARRY)
INPUT GATING
INTERCONNECTION INPUTS

FIG.2

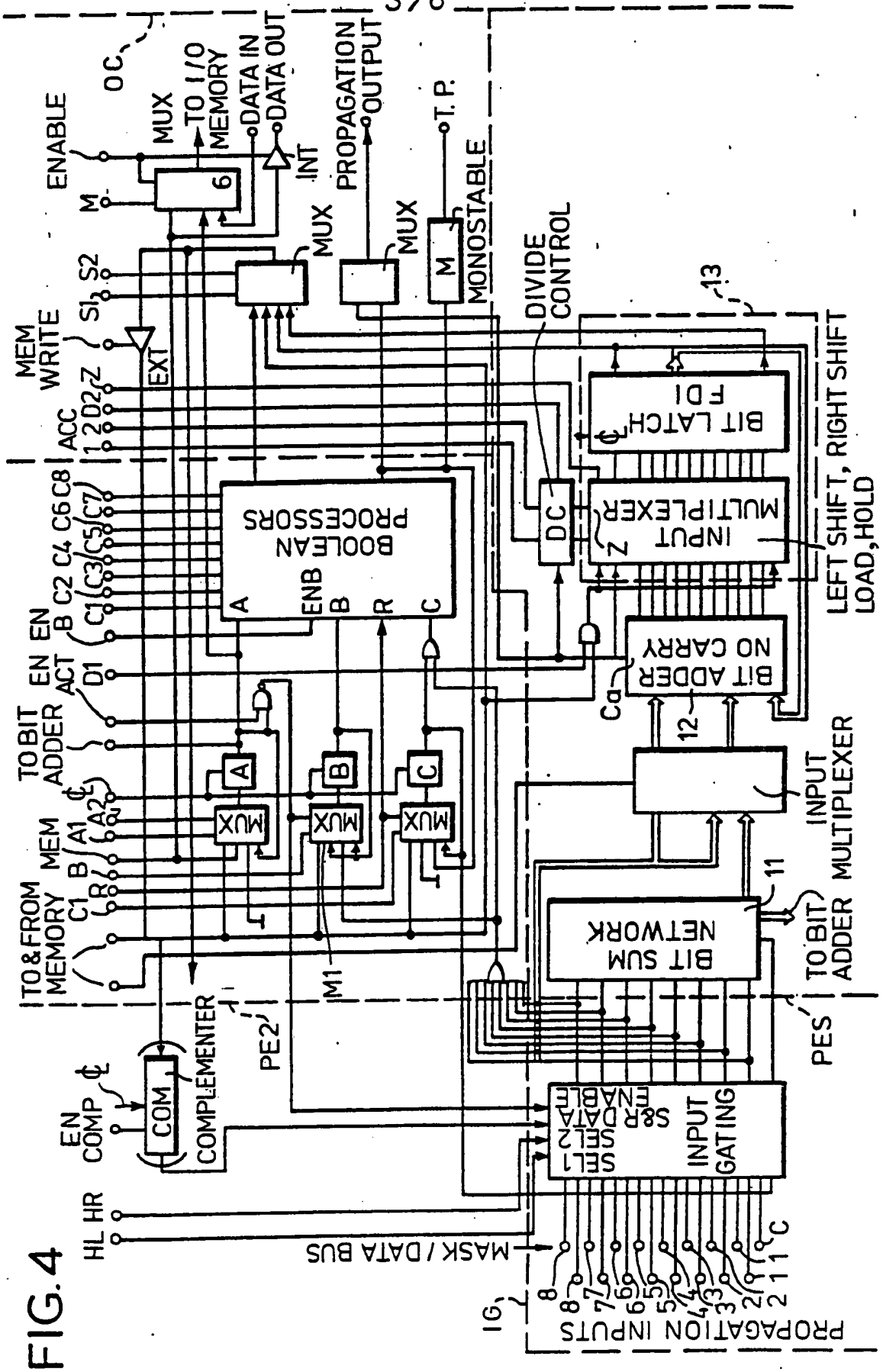

OC
ENABLE
M
MUX
6
TO I/O MEMORY
DATA IN
DATA OUT
INT
PROPAGATION OUTPUT
T.P.
MONOSTABLE
DIVIDE CONTROL
MUX
MUX
S1 S2
MEM WRITE
EXT
ACC 1 2 D2 Z
B C2 C4 C6 C8
C1 C3 C5 C7
EN B
EN ACT
D1
TO BIT ADDER
MEM A1 A2
B
C1 R
TO & FROM MEMORY
BOOLEAN PROCESSORS
A
ENB
B
R
C
DC
BIT LATCH
FDI
INPUT MULTIPLEXER
Z
LEFT SHIFT, RIGHT SHIFT
LOAD, HOLD
BIT ADDER
NO CARRY
Ca
12
INPUT MULTIPLEXER
MUX
MUX
MUX
A
B
C
M1
11
BIT SUM NETWORK
TO BIT ADDER
PE2
PES
COMPLEMENTER
EN COMP
COM
SEL1
SEL2
S&R DATA ENABLE
INPUT GATING
HL HR
MASK / DATA BUS
16
8 7 6 5 4 3 2 1 C
PROPAGATION INPUTS

FIG. 4

FIG.5

GROUP	$x = 0$ to $x = 8$	MAXIMUM VALUE OF GROUP
1	$\Sigma A_x^0 N_x^0$	9
2	$+ 2 (\Sigma A_x^0 N_x^1 + \Sigma A_x^1 N_x^0)$	2×18
3	$+ 4 (\Sigma A_x^0 N_x^2 + \Sigma A_x^1 N_x^1 + \Sigma A_x^2 N_x^0)$	4×27
4	$+ 8 (\Sigma A_x^0 N_x^3 + \Sigma A_x^1 N_x^2 + \Sigma A_x^2 N_x^1 + \Sigma A_x^3 N_x^0)$	8×36
5	$+ 16 (\Sigma A_x^0 N_x^4 + \Sigma A_x^1 N_x^3 + \Sigma A_x^2 N_x^2 + \Sigma A_x^3 N_x^1 + \Sigma A_x^4 N_x^0)$	16×45
6	$+ 32 (\Sigma A_x^0 N_x^8 + \Sigma A_x^1 N_x^4 + \Sigma A_x^2 N_x^3 + \Sigma A_x^3 N_x^2 + \Sigma A_x^4 N_x^1 + \Sigma A_x^5 N_x^0)$	32×54
7	$+ 64 (\Sigma A_x^0 N_x^6 + \Sigma A_x^1 N_x^5 + \Sigma A_x^2 N_x^4 + \Sigma A_x^3 N_x^3 + \Sigma A_x^4 N_x^2 + \Sigma A_x^5 N_x^1 + \Sigma A_x^6 N_x^0)$	64×63
8	$+ 128 (\Sigma A_x^0 N_x^7 + \Sigma A_x^1 N_x^6 + \Sigma A_x^2 N_x^5 + \Sigma A_x^3 N_x^4 + \Sigma A_x^4 N_x^3 + \Sigma A_x^5 N_x^2 + \Sigma A_x^6 N_x^1 + \Sigma A_x^7 N_x^0)$	128×72
9	$+256 (\Sigma A_x^1 N_x^7 + \Sigma A_x^2 N_x^6 + \Sigma A_x^3 N_x^5 + \Sigma A_x^4 N_x^4 + \Sigma A_x^5 N_x^3 + \Sigma A_x^6 N_x^2 + \Sigma A_x^7 N_x^1)$	256×63
10	$+512 (\Sigma A_x^2 N_x^7 + \Sigma A_x^3 N_x^6 + \Sigma A_x^4 N_x^5 + \Sigma A_x^5 N_x^4 + \Sigma A_x^6 N_x^3 + \Sigma A_x^7 N_x^2)$	512×54
11	$+1024 (\Sigma A_x^3 N_x^7 + \Sigma A_x^4 N_x^6 + \Sigma A_x^5 N_x^5 + \Sigma A_x^6 N_x^4 + \Sigma A_x^7 N_x^3)$	1024×45
12	$+2048 (\Sigma A_x^4 N_x^7 + \Sigma A_x^5 N_x^6 + \Sigma A_x^6 N_x^5 + \Sigma A_x^7 N_x^4)$	2048×36
13	$+4096 (\Sigma A_x^5 N_x^7 + \Sigma A_x^6 N_x^6 + \Sigma A_x^7 N_x^5)$	4096×27
14	$+8192 (\Sigma A_x^6 N_x^7 + \Sigma A_x^7 N_x^6)$	8192×18
15	$+16384 (\Sigma A_x^7 N_x^7)$	16384×9

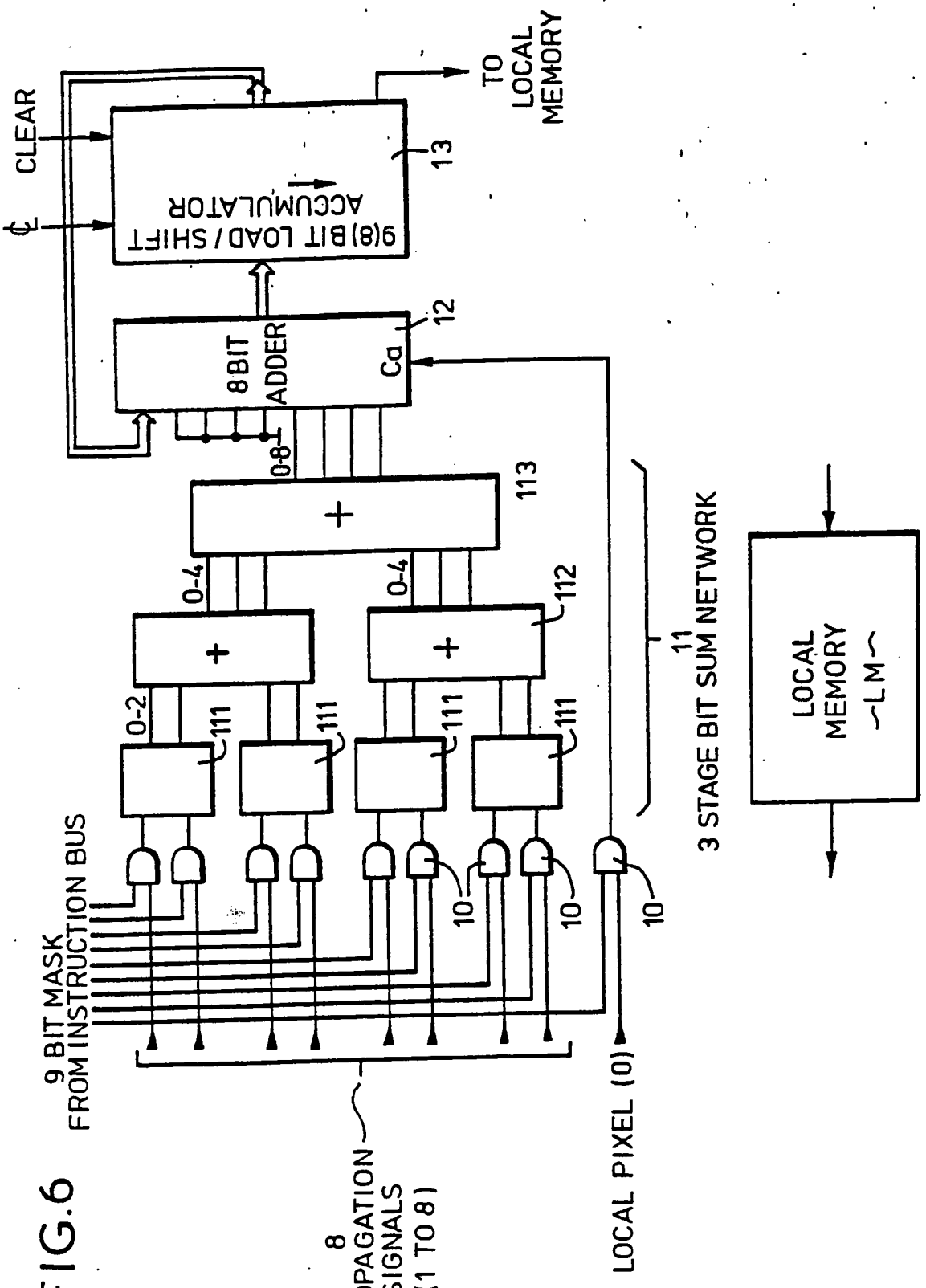

CLEAR
TO LOCAL MEMORY
9(8) BIT LOAD / SHIFT ACCUMULATOR
13
8 BIT ADDER
Ca
12
0-8
113
0-4
0-4
112
0-2
111
111
111
111
11
3 STAGE BIT SUM NETWORK
LOCAL MEMORY ~LM~
9 BIT MASK FROM INSTRUCTION BUS
10
10
10
8 PROPAGATION SIGNALS (1 TO 8)
LOCAL PIXEL (0)

FIG.6

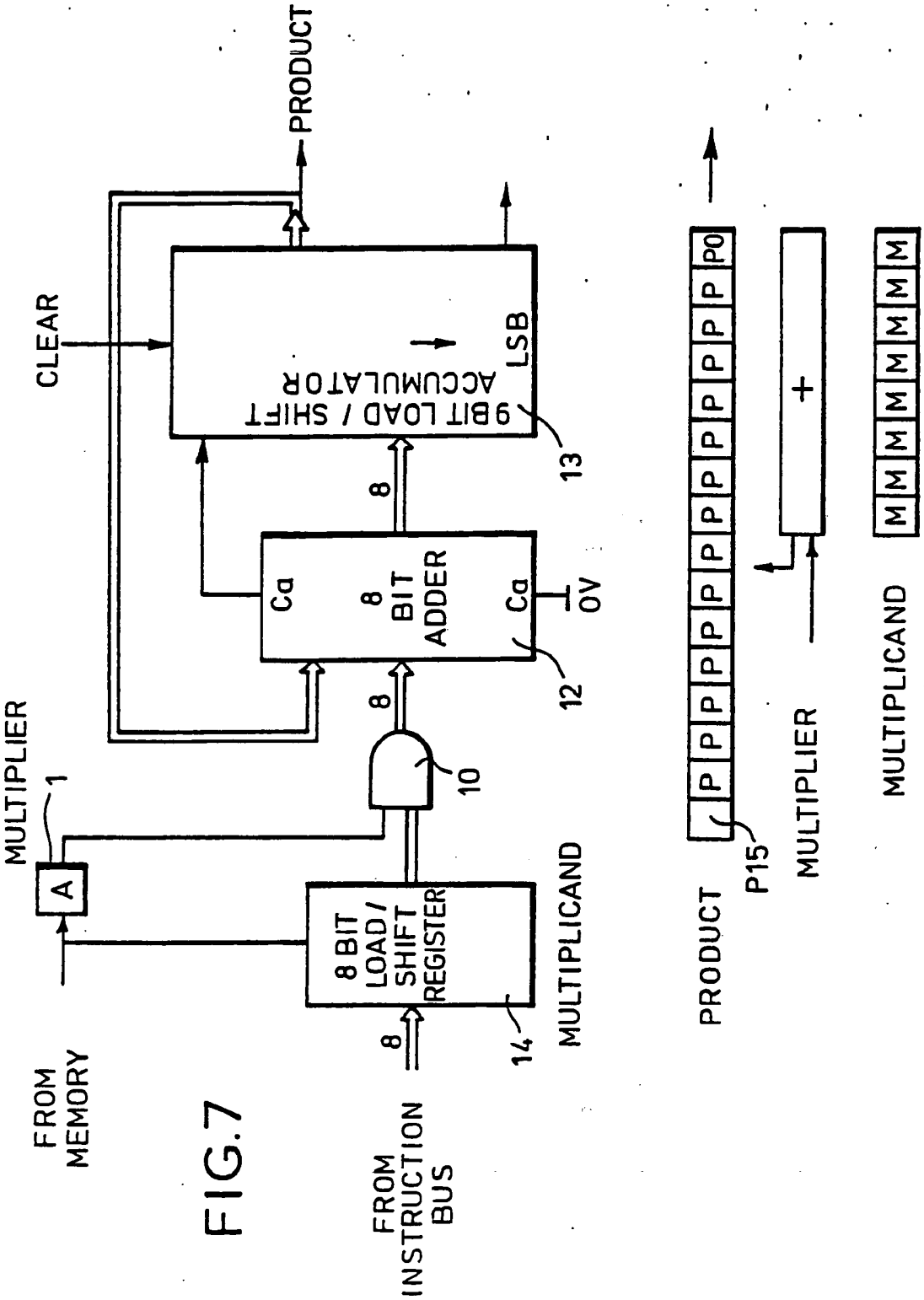
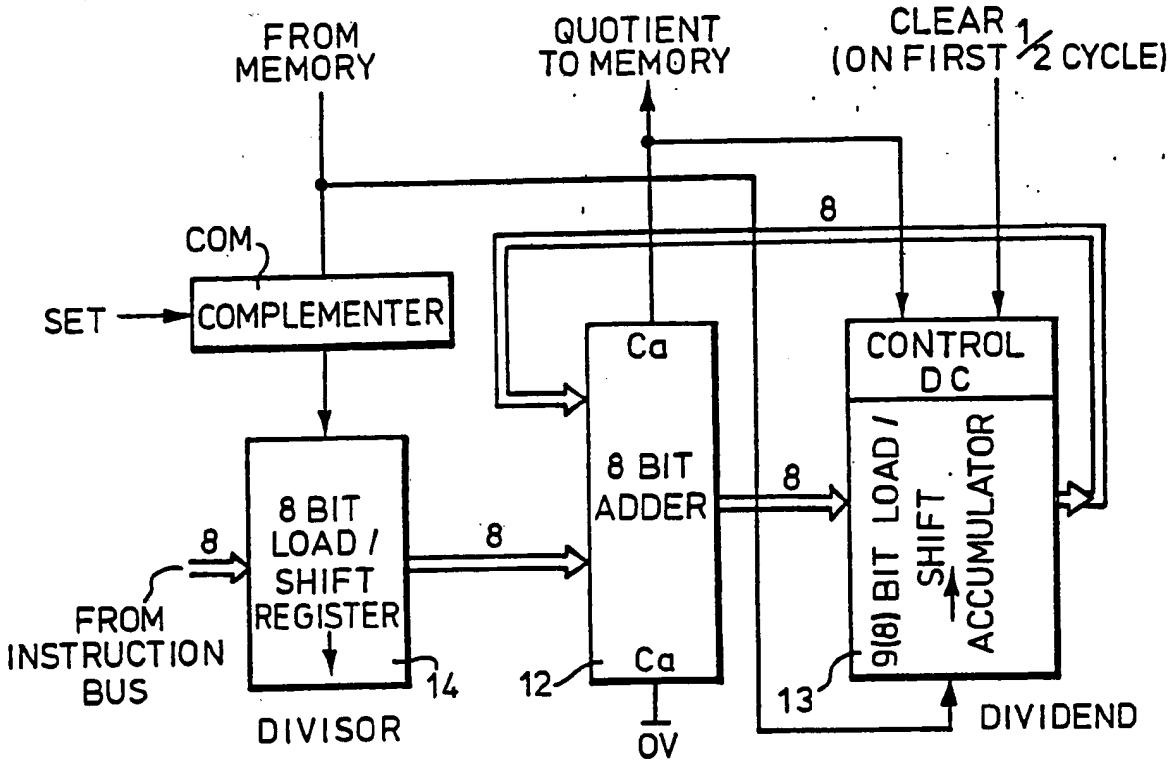

PRODUCT
CLEAR
9 BIT LOAD / SHIFT ACCUMULATOR
LSB
13
8
Ca
8 BIT ADDER
Ca
0V
12
8
10
MULTIPLIER
1
A
8 BIT LOAD / SHIFT REGISTER
MULTIPLICAND
14
8
FROM MEMORY
FROM INSTRUCTION BUS
P P P P P P P P
P0
P15
PRODUCT
+
MULTIPLIER
M M M M M M M M
MULTIPLICAND

FIG.7

FIG.8


FROM MEMORY
QUOTIENT TO MEMORY
CLEAR (ON FIRST 1/2 CYCLE)
8
COM
SET
COMPLEMENTER
Ca
CONTROL DC
8 BIT ADDER
8 BIT LOAD / SHIFT REGISTER
9(8) BIT LOAD / SHIFT ACCUMULATOR
8
8
8
FROM INSTRUCTION BUS
14
12
Ca
13
DIVISOR
OV
DIVIDEND

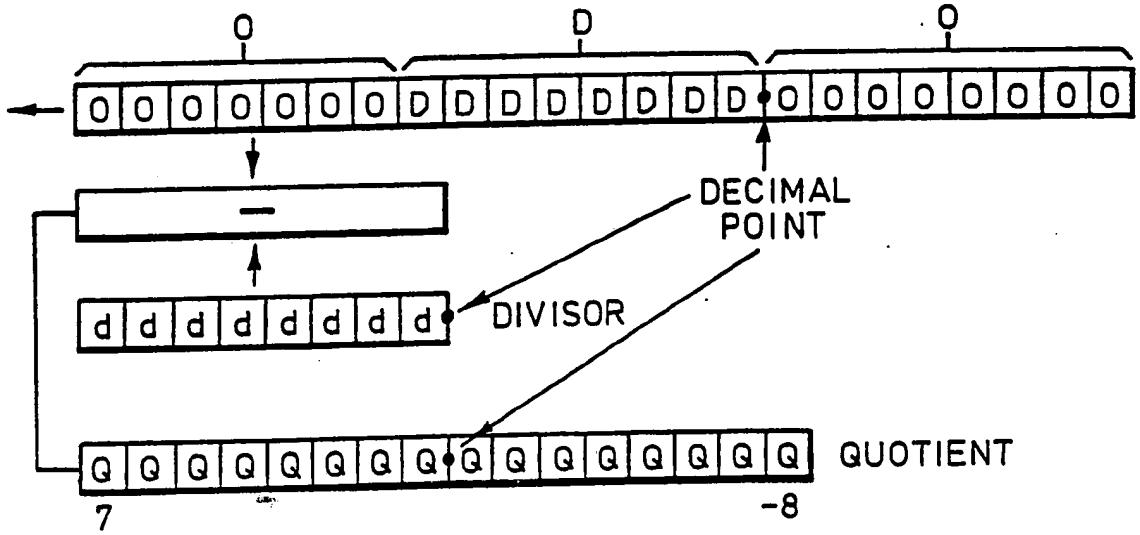

O
D
O
O O O O O O O D D D D D D D D O O O O O O O O
DECIMAL POINT
d d d d d d d d
DIVISOR
Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q
QUOTIENT
7
-8

FIG.9

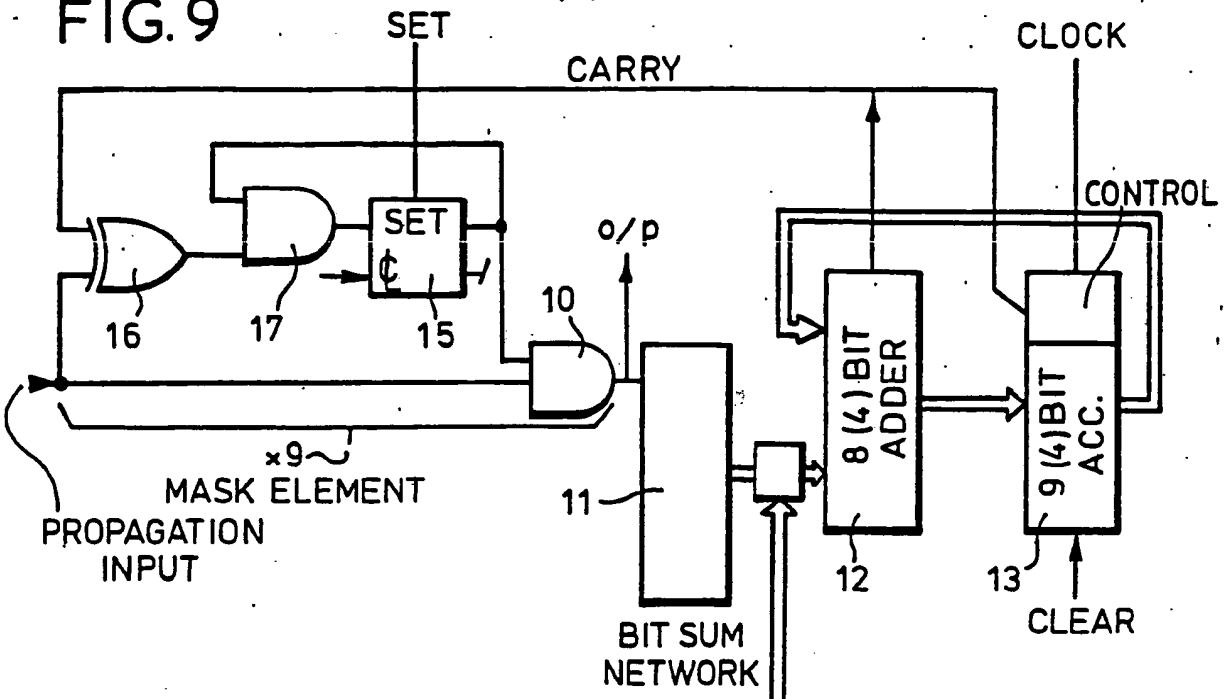
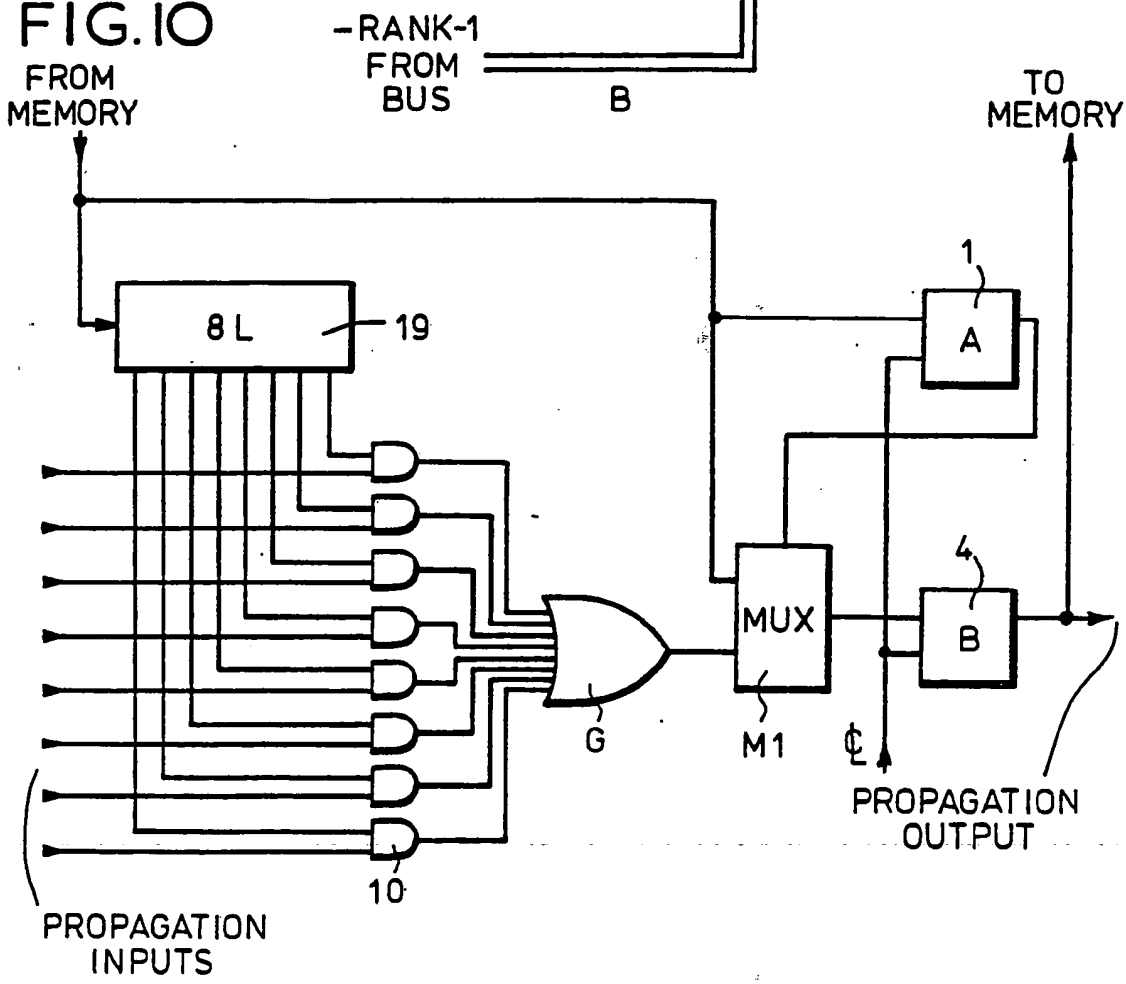

SET
CARRY
CLOCK
CONTROL
SET
o/p
16
17
15
10
8(4)BIT ADDER
9(4)BIT ACC.
×9
MASK ELEMENT
PROPAGATION INPUT
11
12
13
CLEAR
BIT SUM NETWORK

FIG.10


-RANK-1 FROM BUS
B
FROM MEMORY
TO MEMORY
8L
19
1
A
4
B
MUX
G
M1
PROPAGATION OUTPUT
10
PROPAGATION INPUTS